. LABORATOIRE 4 : Héritage – Polymorphisme – Classes abstraites

.Dans tous les exercices suivants, les classes doivent être sous forme canonique.

.

.Projet véhicule :

1.1) Construisez le diagramme de classes des classes décrites ci-dessous.

1.2) Implémentez ces classes, utilisez le programme test pour tester votre implémentation.

Classe véhicule :

- nombrePortes
- nombreCylindres
- couleurVehicule
- niveauCarburant
- typeTransmission
- Vehicule(...)
- getNombreCylindres(...)
- getCouleur(...)
- setCouleur(...)
- getNiveauCarburant(...)
- setNiveauCarburant(...)
- getTypeTransmission(...)
- getNombrePortes(...)
- getClassName(...)
- str(…)
- ...

Classe Taxi :

- passagers
- Taxi(...)
- estOccupe(...)
- setPassagers(...)
- klaxonne(...)
- getClassName(...)
- str(...)

Classe Camion

- charge
- Camion(...)
- estCharge(...)
- setCharge(...)
- klaxonne(...)
- getClassName(...)
- str(...)

Programme test :

```
#include <iostream>
using namespace std;
#include "vehicule.h"
#include "taxi.h"
#include "camion.h"

int main()
{
  Taxi itaxi( 3.3 );

  Camion icamion( 7.54 );
  icamion.setCharge();

  cout<<icamion.str();
  cout<<icamion.klaxonne();

  cout<<itaxi.str();
  cout<<itaxi.klaxonne();

  system("PAUSE");
  return 0;

}
```

.Projet compagnie :

Une compagnie rétribue ses employés sur une base hebdomadaire.

La compagnie dispose de 3 types d'employés :

- employés salariés qui reçoivent un salaire hebdomadaire fixe quel que soit le nombre d'heures presté ;
- employés horaires qui sont payés à l'heure (au-dessus de 40 heures, les heures supplémentaires sont majorées de 50 %) ;
- employés commission qui reçoivent un pourcentage sur leurs ventes ;

Réalisez un diagramme de classes qui modélise la situation.

Réalisez ensuite un programme qui permet de calculer la rétribution de chaque employé selon son type.

Testez votre programme en créant un employé de chaque type :

Type	Prénom	Nom	Salaire	Numéro Sécurité Sociale	Ventes	Commission	Heures prestées
salarié	Jean	Dupont	500 €/semaine	111-11-1111			
horaire	Pierre	Grojean	20 €/heure	222-22-2222			45
commission	Robert	Ledoux		333-33-3333	10 000	6 %	

.
.Programme test:

```
.int main()
.{
.  cout << fixed << setprecision( 2 );
.  vector < Employe * > employees( 4 );
.  employees[ 0 ] = new EmployeSalarie( "Jean", "Dupont","111-11-1111", 500.00 );
.  employees[ 1 ] = new EmployeHoraire( "Pierre", "Grojean","222-22-2222", 20.00,45.00);
.  employees[ 2 ] = new EmployeCommission( "Robert","Ledoux", "333-33-3333", 10000.00,0.06);
.
.  for ( int i = 0; i < employees.size(); i++ )
.  {
.     cout<<*employees[ i ];
.     cout << "Retribution : " << employees[ i ]->gain() << endl;
.  }
.  for ( int j = 0; j < employees.size(); j++ )
.  {
.     cout << "\nsuppression objet "
.          << employees[ j ] ->name();
.      delete employees[ j ];
.
.  }
.  cout << endl;
.  system("PAUSE");
.  return 0;
.}
```

.Projet compagnie : prolongement gestion entreprise

La classe Entreprise est définie par un nom et un vecteur (vector) de pointeurs d'employés gérés par agrégation interne (par pointeurs).
Cette classe est sous forme canonique et dispose au minimum des méthodes ajouterEmploye et str.

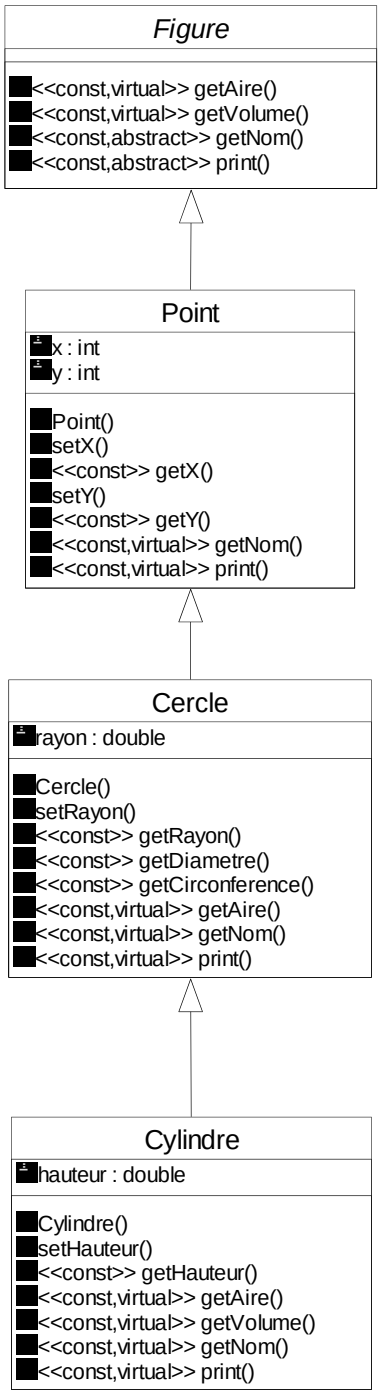
Suggestion : afin de mettre en place le polymorphisme au moment d'ajouter un employé, vous pouvez implémenter une méthode clone.

Code source pour tester votre classe :

```
Entreprise entreprise;
  for ( int i = 0; i < employees.size(); i++ ){
    entreprise.ajouterEmploye(employees[i]);
  }
   for ( int i = 0; i < employees.size(); i++ ){
    entreprise.ajouterEmploye(employees[i]);
  }
  cout<<endl<<"Affichage entreprise AIP"<<endl;
  cout<<entreprise.str();

  vector < Employe * > employees2(4);
  for ( int i = 0; i < employees.size(); i++ ) {
      employees2[ i ]=employees[ i ]->clone(); //verifier affectation
  }

  Entreprise entreprise2;
  for ( int i = 0; i < employees2.size(); i++ ){
    entreprise.ajouterEmploye(employees2[i]);
  }
   for ( int i = 0; i < employees2.size(); i++ ){
    entreprise2.ajouterEmploye(employees2[i]);
  }
  cout<<endl<<"Affichage entreprise AIP 2"<<endl;
  cout<<entreprise.str();

  Entreprise e3;
  cout<<e3.str();
  cout<<"********************************"<<endl;
  e3=entreprise2;
  cout<<e3.str();
```

.Hiérarchie de figures : polymorphisme

Implémentez le diagramme de classes suivant :



Testez votre programme à l'aide du code suivant :

```
void infoViaPointeur( const Figure *fig ) ;
void infoViaReference( const Figure &fig );

...

cout << fixed << setprecision( 2 );
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
   Point point( 1, 5 );                    //point: x=1, y=5
   Cercle cercle( 13, 2, 6.5 );            //cercle: centre(13,2),rayon=6.5
   Cylindre cylindre( 3, 4, 10.0, 1 ); //cylindre:centre base(3,4),rayon=10,hauteur=1
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
   cout << point.getNom() << ": ";
   point.print();

   cout << cercle.getNom() << ": ";
   cercle.print();

   cout << cylindre.getNom() << ": ";
   cylindre.print();

////////////////////////////////////////////////////////////////////////////////////////////////////////////////
   Point *pPoint=&point;
   Cercle *pCercle=&cercle;
   Cylindre *pCylindre=&cylindre;

   infoViaPointeur(pPoint);
   infoViaPointeur(pCercle);
   infoViaPointeur(pCylindre);

   pPoint=pCylindre;
   infoViaPointeur(pPoint);
   infoViaReference(*pPoint);

   pCercle=pCylindre;
   infoViaPointeur(pCercle);
   infoViaReference(*pCercle);
```

Créez ensuite une classe ListeFigures qui contient un vecteur de pointeurs de figures par agrégation interne. Cette classe dispose au minimum des méthodes ajouterFigure, retirerFigure et str.